(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **2000066923 A**

(43) Date of publication of application: **03.03.00**

(51) Int. Cl **G06F 11/28**

(21) Application number: **10239249**

(22) Date of filing: **25.08.98**

(71) Applicant: **NIPPON TELEGR & TELEPH CORP <NTT>**

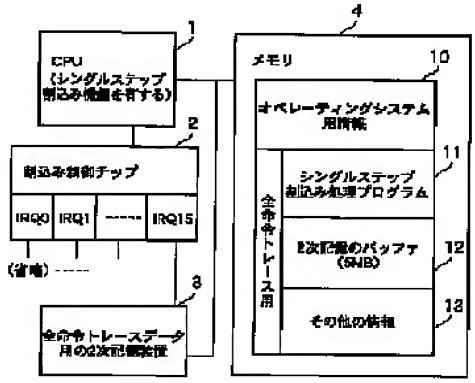
(72) Inventor: **SUGIMURA YASUSHI**

(54) **METHOD FOR CONTROLLING ALL-INSTRUCTION TRACING IN EMULATOR**

(57) Abstract:

PROBLEM TO BE SOLVED: To realize an emulator all-instruction tracing controlling method capable of correctly and accurately executing the tracing of all instructions in an emulator which can not be realized by conventional processing.

SOLUTION: Even when the emulator tries to turn off the trace flag of a CPU 1 during the execution of all-instruction tracing, an all-istruction tracing single step interruption processing program prevents the trial, collects all- instruction tracing data and even when the emulator turns on the tracing flag of the CPU 1, the program turns off the flag and does not collect the all- instruction tracing data.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2000－66923
（P2000－66923A）
(43)公開日　平成12年３月３日(2000.3.3)

(51)Int.Cl.[7]	識別記号	ＦＩ	テーマコード（参考）
Ｇ０６Ｆ　11/28	３１０	Ｇ０６Ｆ　11/28	３１０Ｂ　５Ｂ０４２

審査請求　未請求　請求項の数１　ＯＬ　（全　８　頁）

(21)出願番号　特願平10－239249

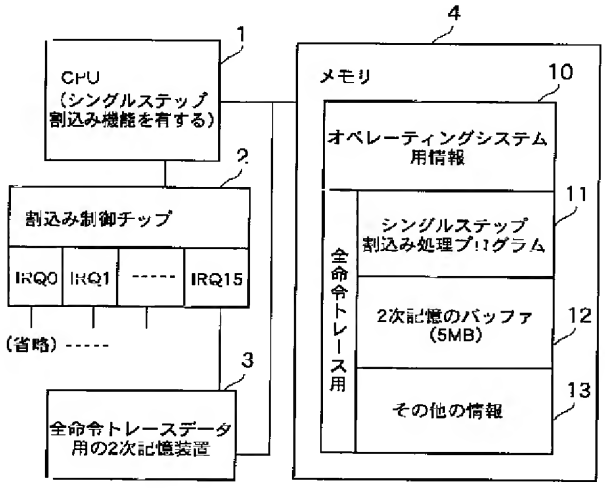
(22)出願日　平成10年８月25日(1998.8.25)

(71)出願人　000004226
日本電信電話株式会社
東京都千代田区大手町二丁目３番１号
(72)発明者　杉村　康
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内
(74)代理人　100083806
弁理士　三好　秀和　（外１名）

Ｆターム(参考)　5B042 GA03 HH23 HH30 JJ40

(54)【発明の名称】　エミュレータにおける全命令トレースの制御方法

(57)【要約】
【課題】　従来の処理では実現が不可能であったエミュレータにおける全命令トレースをすべて正しく適確に行うことができるエミュレータにおける全命令トレースの制御方法を提供する。
【解決手段】　全命令トレース実行中時にエミュレータがＣＰＵのトレースフラグをオフにしようとしても、全命令トレース用シングルステップ割り込み処理プログラムがそれを阻止し、全命令トレースデータの収集を行い、また全命令トレース仮停止中時にエミュレータがＣＰＵのトレースフラグをオンにしても、全命令トレース用シングルステップ割り込み処理プログラムがオフに戻し、この時全命令トレースデータの収集を行わない。



【特許請求の範囲】
【請求項１】　情報処理装置内の中央演算ユニットが全命令トレースを行い、このトレース対象のオペレーティングシステムが他のシステムのシステムコールのエミュレーションを行い、このエミュレーションを含むすべての命令の実行の全命令トレースを行う場合において、全命令トレース用シングルステップ割り込み処理プログラムの外部変数の計数手段としてカウンタを設け、全命令トレースの実行中においては、前記プログラムがエミュレーションにおけるトレースフラグをオフにする命令を検出した場合、このオフを阻止し、前記カウンタの値に１を加算し、全命令トレースの終了以降においては、前記カウンタの値が０でない場合に、前記プログラムがシングルステップ割り込みの発生を監視し、該割り込みが発生した場合は、トレースフラグをオフとし、前記カウンタから１を減算し、この結果のカウンタの値が０でない場合は、前記監視を継続し、この結果のカウンタの値が０である場合は、前記監視を停止して、全命令トレースのすべての動作を終了することを特徴とするエミュレータにおける全命令トレースの制御方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、情報処理装置内の中央演算ユニット（以下、ＣＰＵと略記する）が全命令トレースを行い、このトレース対象のオペレーティングシステムが他のシステムのシステムコールの動作をエミュレートするエミュレータにおける全命令トレースの制御方法に関し、特に電子計算機、ワードプロセッサ、電子交換機、ワークステーション等の情報処理装置内のＣＰＵの全命令トレースの制御方法に関する。
【０００２】
【従来の技術】一般に、性能評価等の目的のため、評価対象のオペレーティングシステムのプログラム（該オペレーティングシステム上で動作するすべてのアプリケーションプログラムを含み、全命令トレースを行うためのプログラムを除く、以下省略）の命令群の内、ＣＰＵが実行するすべての命令に関する情報（実行された命令の内容、命令の長さ、レジスタの変化状況、変化したレジスタの内容、その他必要に応じてプログラムの管理単位であるスレッドの情報等の情報）を１命令の実行直前毎に収集することを、全命令トレースと言い、その全命令トレースによって収集された情報を全命令トレースデータと言う。上記プログラムは、関数（プログラムからコールされ、入り口が１つであるプログラム）並びに割り込み処理プログラム（割り込み等を契機としてコールされるプログラムであり、通常はプログラムからはコールされないプログラム）からなり、全命令トレースにおいてはそれらのすべてがトレース対象である（関数のみのトレースしか行わないものは、全命令トレースとは言わない）。
【０００３】シングルステップ割り込み機能を持つＣＰＵの元での全命令トレースにおいては、関数に関しては予めＣＰＵのトレースフラグをＯＮとしておき、割り込み処理に関しては、その先頭でＣＰＵのトレースフラグをＯＮとする等によって、ＣＰＵに１命令の実行直前毎にシングルステップ割り込みを起こさせることが一般的に行われている。
【０００４】ＣＰＵが評価対象のオペレーティングシステムの１命令を実行する直前毎に、ＣＰＵ自身に起こさせるシングルステップ割り込みを利用して、ＣＰＵは全命令トレース用シングルステップ割り込み処理プログラムに制御を渡し、そのプログラムが全命令トレースデータを収集し、一旦全命令トレースデータバッファというメモリ上のエリアに格納して、上記割り込み元にリターンし、リターン後にＣＰＵは、割り込まれた命令を１命令実行し、その実行が終わって、次の１命令を実行する直前に、上記と同様にシングルステップ割り込みを起こさせることにより、評価対象のオペレーティングシステムのプログラムの命令群の内、実行されるすべての命令の全命令トレースが行われる。
【０００５】上記全命令トレースにおいては、一般に、何らかの手法で全命令トレースの開始点を検出した時点（その手法は多数あり、任意、以下全命令トレース開始時点と略記）で、ＣＰＵのトレースフラグをＯＮとし、何らかの手法で全命令トレースの停止点を検出した時点（その手法は多数あり、任意、以下全命令トレース停止し時点と略記）でＣＰＵのトレースフラグをＯＦＦとすることにより、その時実行中のプログラムの全命令トレースを制御するが、以下に示す手法の中で示すように、それだけでは全命令トレース開始時点～全命令トレース終了時点までのすべてのスレッドや割り込み処理の全命令トレースを行うことができないので、従来より、一般に全命令トレースにおいては下記の２つの手法を併用する。
【０００６】（１）全命令トレースは、割り込み処理においても、全命令トレースデータを収集しなければならないが、割り込み処理においては、一般に、ＣＰＵが、割り込まれた時のトレースフラグをスタックに退避し、トレースフラグをＯＦＦとした後に、割り込み処理プログラムに制御を渡す。従って、何もしないと、割り込み処理プログラムの命令の実行においてシングルステップ割り込みが発生しないため、全命令トレースデータを収集することができない。それに対処するために、上記全命令トレース開始時点で、外部変数「全命令トレース実行中」に１（初期値は０）を設定し、上記全命令トレース停止時点で、該外部変数の値を０に戻し、割り込み処理の先頭で、該外部変数の値をチェックし、該値が１の時のみ、ＣＰＵのトレースフラグに１を設定することにより、全命令トレース実行中にのみ、割り込み処理の部分も全命令トレースを行う（当然ながら、シングルステ

ップ割り込みは、全命令トレースの情報を収集する処理であるので、上記処理は行わない。シングルステップ割り込み処理だけは、全命令トレース対象外である）。

【０００７】（２）一般にオペレーティングシステム内にはスレッド（独立して走行するプログラムの単位）が多数存在し、オペレーティングシステムは、それらのスレッドを適宜切り換えて実行することにより、ＣＰＵの有効利用を図る。しかし、ＣＰＵが１時点で実行しているスレッドは、ただ１つである。そのため、その時実行されていないスレッドは、その実行は一旦中断されている。そして、その中断されたスレッドを適宜再開するために、中断時点のレジスタ等の情報を、そのスレッドのスタックエリアに退避している。その中には、ＣＰＵのトレースフラグも含まれる。従って、何もしないと、現在のスレッドの実行が中断され、既に中断されているスレッドを再開する時に、スタックエリアに退避されているトレースフラグ等は、中断直前の状態に回復され、その時に、トレースフラグがＯＦＦになってしまうため、全命令トレースが行われない結果となる。それらを回避する（すべてのスレッドの全命令トレースを実行させる）ために、一般に、上記トレース開始時点において、すべてのスレッドのスタックエリアに既に退避されているトレースフラグをすべてＯＮにし、上記トレース停止時点において、ＯＦＦに戻すことにより、全命令トレース実行中に、すべてのスレッドの全命令トレースが行われるようにしている。上記スタックエリアは、一般にカーネルスタックエリアと呼ばれ、そのスタックエリアは、カーネル空間（全命令トレース用シングルステップ割り込み処理プログラム等のカーネル関連プログラムから可視である空間）にあり、スレッド毎に予め決められた構造体内にあるので、上記のように、トレース開始時点とトレース停止時点に、一度に変更することが可能である。

【０００８】

【発明が解決しようとする課題】しかしながら、最近においては、１つのオペレーティングシステム内において別のオペレーティングシステムのシステムコールの動作を、ユーザ空間上でエミュレートするエミュレータが実現されている。該エミュレータは、オペレーティングシステムが動作するカーネル空間で動作するのではなく、スレッド等毎にあるユーザ空間上で動作し、そのため、そのスタック情報は、ユーザスタック（上記カーネルスタックとは別のユーザ毎のスタック）に格納される。

【０００９】一方、他のオペレーティングシステムのシステムコールをユーザ空間上でエミュレーションするためには、システムコールで呼ばれるプログラムはシステム空間になければならないので、システムコールで一旦システム空間に行った時に、そのシステムコールルーチン等によりスタック等に退避された情報を、ユーザスタック上にコピーし、その後、ユーザ空間にジャンプして、システムコールのエミュレート動作を行い、最後に、㋫上記ユーザスタック上にコピーされた情報を回復することによって、上記システムコールの発行元にリターンする。そしてその回復される情報の中に、トレースフラグが含まれる。即ち、そのトレースフラグの情報は、ユーザスタック内にあり、ユーザスタック内のどの位置にあるかは、そのスレッドでどのようにユーザスタックが使用されたかに依存し（上記のカーネルスタックのように、スレッド毎の特定の位置にある訳ではないので）、従来の技術で述べた手法のように、トレース開始時点やトレース終了時点では、上記のトレースフラグのアドレスを特定することができないので、変更することができない（そのスタックを実際に退避回復する場所でしか、その位置を知ることができない）。

【００１０】従って、従来の手法においては、別のオペレーティングシステムのシステムコールの動作をユーザ空間上でエミュレートするエミュレータが搭載されているシステムにおいては、上記の㋫のトレースフラグの回復が行われた時点以降はトレースフラグがＯＦＦになってしまい、そのため、そのエミュレータが動作しているスレッドの全命令トレースを行うことができないという問題がある。

【００１１】本発明は、上記に鑑みてなされたもので、その目的とするところは、従来の処理では実現が不可能であったエミュレータにおける全命令トレースをすべて正しく適確に行うことができるエミュレータにおける全命令トレースの制御方法を提供することにある。

【００１２】

【課題を解決するための手段】上記目的を達成するため、請求項１記載の本発明は、情報処理装置内の中央演算ユニットが全命令トレースを行い、このトレース対象のオペレーティングシステムが他のシステムのシステムコールのエミュレーションを行い、このエミュレーションを含むすべての命令の実行の全命令トレースを行う場合において、全命令トレース用シングルステップ割り込み処理プログラムの外部変数の計数手段としてカウンタを設け、全命令トレースの実行中においては、前記プログラムがエミュレーションにおけるトレースフラグをオフにする命令を検出した場合、このオフを阻止し、前記カウンタの値に１を加算し、全命令トレースの終了以降（以下、全命令トレース仮停止中と略記）においては、前記カウンタの値が０でない場合に、前記プログラムがシングルステップ割り込みの発生を監視し、該割り込みが発生した場合は、トレースフラグをオフとし、前記カウンタから１を減算し、この結果のカウンタの値が０でない場合は、前記監視を継続し、この結果のカウンタの値が０である場合は、前記監視を停止して、全命令トレースのすべての動作を終了することを要旨とする。

【００１３】請求項１記載の本発明にあっては、全命令トレース実行中時にエミュレータがＣＰＵのトレースフ

ラグをオフにしようとしても、全命令トレース用シングルステップ割り込み処理プログラムがそれを阻止し、全命令トレースデータの収集を行い、また全命令トレース仮停止中時にエミュレータがＣＰＵのトレースフラグをオンにしても、全命令トレース用シングルステップ割り込み処理プログラムがオフに戻し、この時全命令トレースデータの収集を行わないため、エミュレータにおける全命令トレースを適確に行うことができる。

【００１４】

【発明の実施の形態】本発明の実施形態を説明する前に、まず上述したように、従来、エミュレータが動作しているスレッドの全命令トレースを行うことができないという問題を解決するために利用する４つの事実および本発明の概要について説明する。この４つの事実は以下の通りである。

【００１５】❶ 上記トレースフラグの回復は、ＰＯＰＦＤという特定の命令によってのみ可能であり、エミュレータは上記命令によりトレースフラグを回復する。

【００１６】❷ 上記の特定の命令においては、トレースフラグに設定される値は、（命令仕様に基づいて）ＣＰＵのスタックポインタが示すアドレスの位置に存在する。

【００１７】❸ 上記❷のスタックポインタの内容は、上記命令を実行する直前でシングルステップの割り込みが発生した場合、ＣＰＵの機能により、スタック内に格納される。

【００１８】❹ 従来の手法においては、上記の特定の命令の実行の直前のシングルステップ割り込みまでは、全命令トレースが正常に行われる（上記命令の次の命令以降からシングルステップ割り込みが起こらなくなる）。

【００１９】上記の事実にしたがって、全命令トレース開始時点から全命令トレース仮停止時点の間（以下、全命令トレース実行中時と略記）において以下を行う。

【００２０】（１）命令が実行される直前に発生するシングルステップ割り込みによって制御が渡される「全命令トレース用シングルステップ割り込み処理プログラム」は、全命令トレース実行中時においては、その時に実行されようとした命令が、上記ＰＯＰＦＤであるかどうかをチェックし、上記の特定の命令でないなら、従来と同じ処理を行う。上記の特定の命令であれば、従来の処理を行う前に、以下を実施する。

【００２１】(A) 「”ＣＰＵによって退避されたスタックポインタ”の内容が示すアドレスのエリアの内容」（即ちトレースフラグが格納されているエリアの内容）をチェックし、

(a) トレースフラグがＯＮであるなら、そのエリアの内容のトレースフラグをＯＮとする。これによって、全命令トレース用シングルステップ割り込み処理プログラムからリターンした時に実行されるＰＯＰＦＤによってＣＰＵのトレースフラグがＯＮに設定される。

(b) トレースフラグがＯＦＦであるなら、何もしない。

以上によって、エミュレータを含むすべての全命令トレースが可能となる。

【００２２】しかしながら、上記の実現だけでは、次の問題が発生する。

【００２３】上記（Ａ）によってＯＮが設定されたトレースフラグは、エミュレータによってスタックに退避される場合があるので、全命令トレース停止時に従来の方法によって、すべてのスレッドのトレースフラグをＯＦＦにしても、その後に、上記で退避されていたトレースフラグが回復された時点で、再び、シングルステップ割り込みが発生してしまい、何もしないと再び全命令トレースが実施されてしまう。

【００２４】それらに対処するために、以下を行う。

【００２５】（２）外部変化「トレースＯＮ化累積カウンタ」「全命令トレース仮停止」を設け、予め０に初期設定しておく。

【００２６】（３）上記（１）（Ａ）（ａ）で、「トレースＯＮ化累積カウンタ」に１を加算する。

【００２７】（４）全命令トレース仮停止時に、「トレースＯＮ化累積カウンタ」が０であるかどうかをチェックし、０でないなら、「全命令トレース実行中」に０を設定して、従来の全命令トレースの停止を行うだけでなく、「全命令トレース仮停止」に１を設定する。

【００２８】０なら、「全命令トレース実行中」に０を設定して、従来の全命令トレースの停止を行う。

【００２９】（５）全命令トレース用シングルステップ割り込み処理プログラムは、「全命令トレース仮停止」が１であるかどうかをチェックし、０であるなら、従来の処理と、上記（１）の処理を行うが、１である場合は、従来の処理も、上記（１）も行わず（全命令トレースデータの収集等は行わず）以下を行う。

【００３０】(a) スタックエリアに退避されているエラーフラグの内容をＯＦＦとすることにより、全命令トレース用シングルステップ割り込み処理プログラムからリターンした時に、ＣＰＵのトレースフラグをＯＦＦとさせる。

(b) 「トレースＯＮ化累積カウンタ」より１を減算する。

(c) 上記（ｂ）の結果が０であるなら、「全命令トレース仮停止」に０を設定することにより、全命令トレースが完全に停止する。

(d) 全命令トレース用シングルステップ割り込み処理プログラムよりリターンする。

【００３１】以上により、オペレーティングシステム内に別のオペレーティングシステムのシステムコールの動作をユーザ空間上でエミュレートするエミュレータが搭載されていても、エミュレータの動作を含んだ全命令トレースを正しく行うことが可能となる。

【００３２】本発明においては、上述したように、オペ

レーティングシステム内に別のオペレーティングシステムのシステムコールの動作をユーザ空間上でエミュレートするエミュレータが搭載されている場合に、全命令トレース実行中時の場合は、エミュレータがトレースフラグをＯＦＦにしようとしても、それを実施する命令の実行の直前で、全命令トレース用シングルステップ割り込み処理プログラムがそれを阻止し、また全命令トレース仮停止時の後に、エミュレータがトレースフラグをＯＮにしても、そのＯＮによる（次の命令が実行される直前に発生する）シングルステップ割り込みによる全命令トレース用シングルステップ割り込み処理プログラムの走行で、全命令トレースデータを収集せず、且つ、該割り込み処理プログラムからリターンした時に、トレースフラグがＯＦＦになるように、スタックの内容を変更するので、エミュレータの動作を含んだ全命令トレースを正しく行うことが可能となる。

【００３３】次に、図面を用いて、本発明の一実施形態に係るエミュレータにおける全命令トレースの制御方法について説明する。図１は、本発明の一実施形態に係るエミュレータにおける全命令トレースの制御方法を実施する装置の構成を示すブロック図である。同図に示す実施形態は、シングルステップ割り込み機能を有するＣＰＵ１（例えば、INTEL Pentium ）、割り込み制御チップ２（例えば、INTEL 8259）、全命令トレースデータ用の２次記憶装置３（例えば、ＩＤＥセカンダリーＨＤ、ＩＲＱ番号１５配下）、およびメモリ４を有する。メモリ４は、オペレーティングシステム用情報１０、全命令トレース用シングルステップ割り込み処理プログラム１１、全命令トレース用２次記憶装置のバッファ１２、および全命令トレース用のその他の情報１３を記憶している。

【００３４】まず、シングルステップ割り込み機能を有するＣＰＵ１は、割り込み制御チップ２と信号線によって接続されており、該チップからの割り込みの有無や該チップの状態の情報等を受け取ることができ、また、該チップへの割り込み受け取り完了の通知等の情報を出力できる。また、ＣＰＵ１は、全命令トレースデータ用の２次記憶装置３およびメモリ４とバスによって接続されており、それらの装置並びにメモリに対する情報のリードおよびライトが可能である。メモリ４内の情報は、オペレーティングシステム用情報１０、全命令トレース用シングルステップ割り込み処理プログラム１１、２次記憶装置のバッファ１２、およびその他の情報１３より構成される。

【００３５】図２は、オペレーティングシステム用情報１０の中にある全命令トレース用シングルステップ割り込み処理プログラム１１の走行時のスタックの構造であり、シングルステップ割り込みによって作成されるスタック５０と、その割り込みが発生する直前のユーザの状態フラグ６０、並びに、スタックポインタ７０からなる。また、スタック５０は、旧スタックセグメント番号５１、旧スタックポインタ５２、旧状態フラグ５３、旧コードセグメント番号５４、旧命令アドレス５５、レジスタ情報群５６からなる。

【００３６】それらの内、スタックポインタ７０は、レジスタ情報群５６の最後の４バイトのアドレスをポイントしており、旧スタックポインタ５２は、ユーザの状態フラグ６０をポイントしている関係を示す。

【００３７】図３は、状態フラグ９０の構造を示しており、該状態フラグは、３２ビットの情報からなり、その内８番目のビットがトレースフラグ９１であることを示す。この状態フラグ９０の構造は、ユーザの状態フラグ６０および旧状態フラグ５３の中の構造を示す。

【００３８】次に、図４および図５に示すフローチャートを参照して、上記実施形態の作用について説明する。なお、この説明では、全命令トレース実行中時においてエミュレータが初めてＰＯＰＦＤ命令を実行しようとしている時を例にとって説明する。

【００３９】この時、外部変数「全命令トレース実行中」の値は１、「トレースＯＮ化累積カウンタ」および「全命令トレース仮停止」の値は０である（外部変数とは、どこからでも見える領域であり、構成上はどこに存在してもよい。本実施形態においては、図１のブロック１３で示すその他の情報の中にある）。

【００４０】全命令トレース実行中時において、エミュレータがＰＯＰＦＤ命令を実行しようとすると、その命令の実行の直前に、シングルステップ割り込みが発生し、ＣＰＵ１は、図４の全命令トレース用シングルステップ割り込み処理プログラム１１のステップＳ１００に制御を渡す。その時、スタック５０には、図２の５１～５５の情報を、ＣＰＵ１のスタックポインタには旧命令アドレス５５のエリアのアドレスを、ＣＰＵ１は、自動的に設定する。次に、ＣＰＵ１は図４のレジスタ等の退避（ステップＳ１００）を実行すると、スタック５０の中のレジスタ情報群５６にレジスタ等の内容を退避し、ＣＰＵ１のスタックポインタ７０の内容は、図２に示すように、レジスタ情報群５６の最後の４バイトの位置のアドレスとなる。

【００４１】次に、外部変数「全命令トレース仮停止」の値が０かどうかをチェックし（ステップＳ１０１）、ここではＹＥＳであるので、ステップＳ１０２へ行き、外部変数「全命令トレース実行中」が０でないかどうかをチェックする（ステップＳ１０２）。ここでは、全命令トレース実行中であり、ＹＥＳであるので、全命令トレースデータを収集してバッファに格納し、適宜、２次記憶装置３に出力する（ステップＳ１０４）。それから、シングルステップ割り込みを起こした命令のアドレスがユーザ空間であるかどうかを［スタック５０内の旧命令アドレス（この場合、上記ＰＯＰＦＤ命令のアドレス）がユーザ空間（本実施形態では、ユーザ空間のアド

レスは値が0XC0000000より小）であるか］チェックする（ステップＳ１０５）。該命令はエミュレータの命令であるから、ユーザ空間であるので、ＹＥＳとなり、次に、該命令がＰＯＰＦＤの命令であるかどうかを［スタック５０内の旧命令アドレス５５を当プログラムが動作しているシステム空間の値に変換して、そのアドレスがポイントするエリアの内容が、0X9D（ＰＯＰＦＤの命令コード）であるかを］チェックし（ステップＳ１０６）、この場合ＹＥＳであるので、ステップＳ１０７で、旧スタックポインタ５２の内容よりユーザの状態フラグ６０のアドレスを得、状態フラグ６０の内、図３のトレースフラグ９１を得、その値がＯＦＦであるかどうかをチェックする（ステップＳ１０８）。ここでは、全命令トレース実行中時でエミュレータが初めてＰＯＰＦＤを実行しようとしている時であるので、該ビットはＯＦＦであり、ＹＥＳとなり、ステップＳ１０９へ行き、ユーザの状態フラグ６０のトレースフラグ９１に１（ＯＮ）を設定し、外部変数「トレースＯＮ化累積カウンタ」に１を加算（結果は１となる）した後、ステップＳ２００へ進む。

【００４２】ステップＳ２００では、（一般にオペレーティングシステム用情報１０内の現スレッドアドレスと、予めユーザが指定した全命令トレースの終了用スレッドアドレスとが一致し、且つスタック５０内の旧命令アドレス５５と、予めユーザが指定した全命令トレースの終了用命令アドレスとが一致するかどうかをチェックすることにより）全命令トレースの終了が必要かどうかをチェックし、（エミュレータ内でトレースを終了することは一般にないので）この場合ＮＯであるので、ステップＳ２０５に行き、レジスタ等の回復を行い、ステップＳ２０６で、ＩＲＥＴ命令で割り込み元に戻る。

【００４３】割り込み元は、上記ＰＯＰＦＤの命令であるから、その命令が実行されるが、そのときユーザの状態フラグ６０のトレースフラグ９１は、ステップＳ１０９でＯＮに変更されているので、上記命令の実行により、ＣＰＵのトレースフラグはＯＮになり、次の命令の実行直前以降も、シングルステップ割り込みは継続して発生することとなる。

【００４４】次に、全命令トレースの終了が必要な命令の実行の直前に発生するシングルステップ割り込みの場合を考える。上記と同様に全命令トレース用シングルステップ割り込み処理プログラムに制御を渡されると、レジスタ等の退避を行い（ステップＳ１００）、全命令トレース仮停止＝０かどうかをチェックし（ステップＳ１０１）、全命令トレース実行中が０でないかどうかをチェックし（ステップＳ１０２）、全命令トレースデータを収集して、適宜２次記憶装置３に出力し（ステップＳ１０４）、シングルステップ割り込みを起こした命令のアドレスがユーザ空間かどうかをチェックし（ステップＳ１０５）、（一般に、全命令トレースの停止は、ユーザ空間で行うので）シングルステップ割り込みを起こした命令はＰＯＰＦＤかどうかをチェックする（ステップＳ１０６）が、（エミュレータ内でトレースを止めることは通常有り得ないので）ＮＯとなり、ステップＳ２００で（一般にオペレーティングシステム用情報１０内の現スレッドアドレスと、予めユーザが指定した全命令トレースの終了用スレッドアドレスとが一致し、且つスタック５０内の旧命令アドレス５５と、予めユーザが指定した全命令トレースの終了用命令アドレスとが一致するかどうかをチェックすることにより）全命令トレースの終了が必要かのチェックを行い、ここでＹＥＳとなり、ステップＳ２０１へ進む。

【００４５】ステップＳ２０１では、従来処理のトレース停止処理（一般にＣＰＵが退避したスタック５０内の旧状態フラグ５３のトレースフラグ９１をＯＦＦとし、オペレーティングシステム用情報１０内にあるすべてのスレッドのカーネルスタック上のトレースフラグ９１をＯＦＦとし、外部変数「全命令トレース実行中」に０を設定する処理）を行い、ステップＳ２０２で、外部変数「トレースＯＮ化累積カウンタ」が０かどうかをチェックする。０である場合は、従来の処理と同様に、ステップＳ２０５に行くことにより、全命令トレースを停止するが、この場合、外部変数「トレースＯＮ化累積カウンタ」は、１であるので、ステップＳ２０４へ行き、外部変数「全命令トレース仮停止」に１を設定し、ステップＳ２０５でレジスタ等の回復を行い、ステップＳ２０６でＩＲＥＴ命令を発行し、割り込み元へ戻る。

【００４６】割り込み元に戻る時に、ＣＰＵは、自動的にスタック５０内の旧状態フラグ５３のトレースフラグ９１をＣＰＵのトレースフラグに設定し、その値は、ステップＳ２０１でＯＦＦに設定されているので、戻った後、次の命令の実行直前等では、シングルステップ割り込みは発生しなくなる。しかし、エミュレータが、ＰＯＰＦＤ命令を実行すると、その時のユーザの状態フラグ６０のトレースフラグ９１は、上記ステップＳ１０９でＯＮに設定したままであるので、該命令を実行した後に、ＣＰＵのトレースフラグはＯＮに戻るので、その後の命令の実行直前にシングルステップ割り込みが発生し、再び、図４の全命令トレース用シングルステップ割り込み処理プログラムのステップＳ１００に制御が渡る。その後、上記と同様に、ステップＳ１００でレジスタ等の退避を行い、ステップＳ１０１で、外部変数「全命令トレース仮停止」が０であるかどうかをチェックするが、該変数には１が設定されているので、ステップＳ１０４の全命令トレースデータの収集等は行われず、ステップＳ１１０へ進む。

【００４７】ステップＳ１１０では、（旧状態フラグ５３の中のトレースフラグ９１をＯＦＦに設定することにより）ＣＰＵが退避したスタック５０内のトレースフラグ９１をＯＦＦに設定し、外部変数「トレースＯＮ化累

積カウンタ」より１を減算し、この場合結果は０となるので、ステップＳ１１１においてＹＥＳとなり、ステップＳ２０３に進み、外部変数「全命令トレース仮停止」に０を設定し、レジスタ等の回復をステップＳ２０５で行い、ステップＳ２０６でＩＲＥＴ命令で割り込み元に戻る。その時、ステップＳ２０５で回復したトレースフラグはＯＦＦに設定されているので、ＣＰＵのトレースフラグは０となり、以降、シングルステップ割り込みは発生しなくなる。また、外部変数「全命令トレース仮停止」と「全命令トレース実行中」の内容は、共に０であるので、以降例えシングルステップ割り込みが発生しても、それは全命令トレースとは関係がないのでステップＳ１０２においてオペレーティングシステムのデバッガへ行くことになる。

【００４８】

【発明の効果】以上説明したように、本発明によれば、オペレーティングシステム内に別のオペレーティングシステムのシステムコールの動作をユーザ空間上でエミュレートするエミュレータが搭載されている場合に、全命令トレース実行中時に、エミュレータがＣＰＵのトレースフラグをＯＦＦにしようとしても、全命令トレース用シングルステップ割り込み処理プログラムが、それを阻止して、且つ全命令トレースデータの収集を行い、また、全命令トレース仮停止中時に、エミュレータがＣＰＵのトレースフラグをＯＮにしても、全命令トレース用シングルステップ割り込み処理プログラムが、それをＯＦＦに戻し、且つ、その時全命令トレースデータの収集を行わないので、従来処理では実現が不可能であったエミュレータにおける全命令トレースをすべて正しく行うことが可能となる。

【図面の簡単な説明】

【図１】本発明の一実施形態に係るエミュレータにおける全命令トレースの制御方法を実施する装置の構成を示すブロック図である。

【図２】図１に示す実施形態における全命令トレース用シングルステップ割り込み処理プログラムの走行時のスタックの構造を示す図である。

【図３】図１に示す実施形態における状態フラグの構造を示す図である。

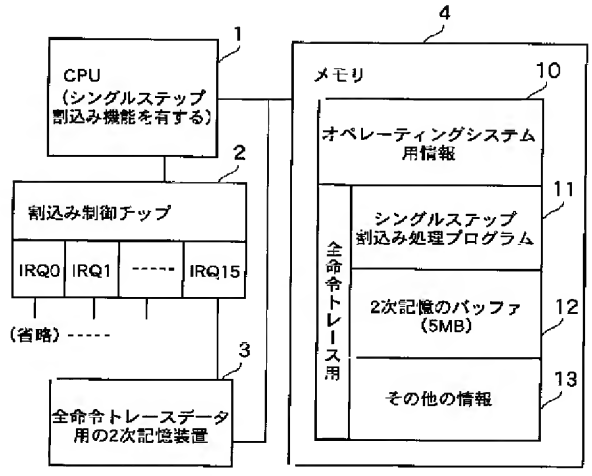
【図４】図１に示す実施形態の作用の一部を示すフローチャートである。

【図５】図１に示す実施形態の作用の図４に続く一部を示すフローチャートである。

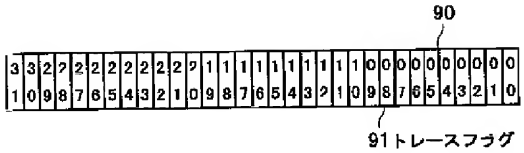
【符号の説明】

１　シングルステップ割り込み機能を有するＣＰＵ
２　割り込み制御チップ
３　全命令トレースデータ用の２次記憶装置
４　メモリ
１０　オペレーティングシステム用情報
１１　全命令トレース用シングルステップ割り込み処理プログラム
１２　全命令トレース用２次記憶装置のバッファ
１３　全命令トレース用のその他の情報
５０　全命令トレース用シングルステップ割り込み処理プログラムの走行時のスタック
５１　旧スタックセグメント番号
５２　旧スタックポインタ
５３　旧状態フラグ
５４　旧コードセグメント番号
５５　旧命令アドレス
５６　レジスタ情報群
６０　ユーザの状態フラグ
７０　スタックポインタ
９０　状態フラグの構造
９１　トレースフラグ

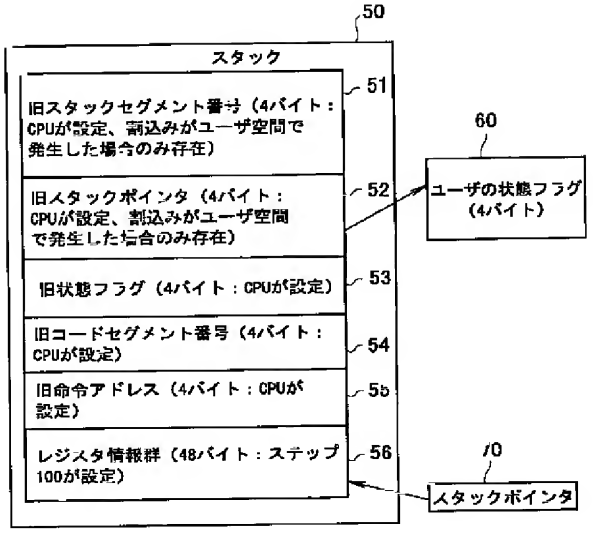
【図１】



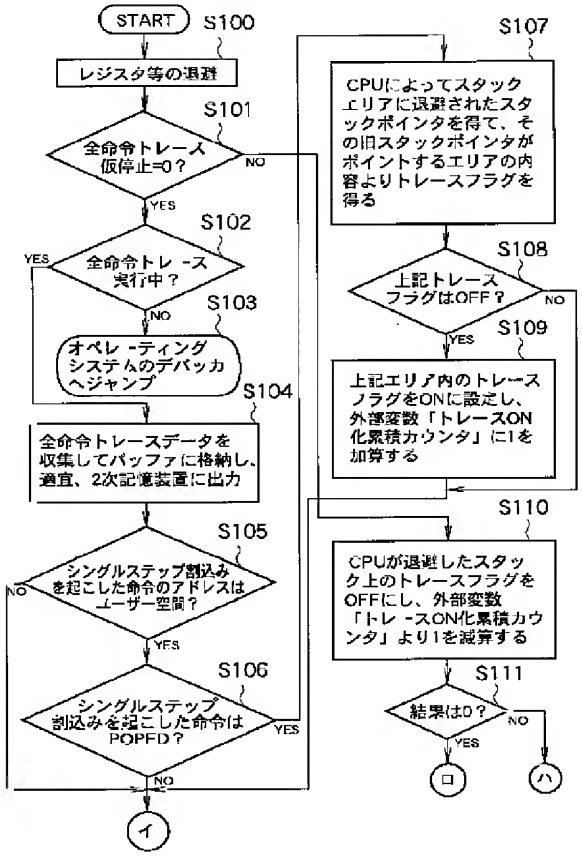
【図３】



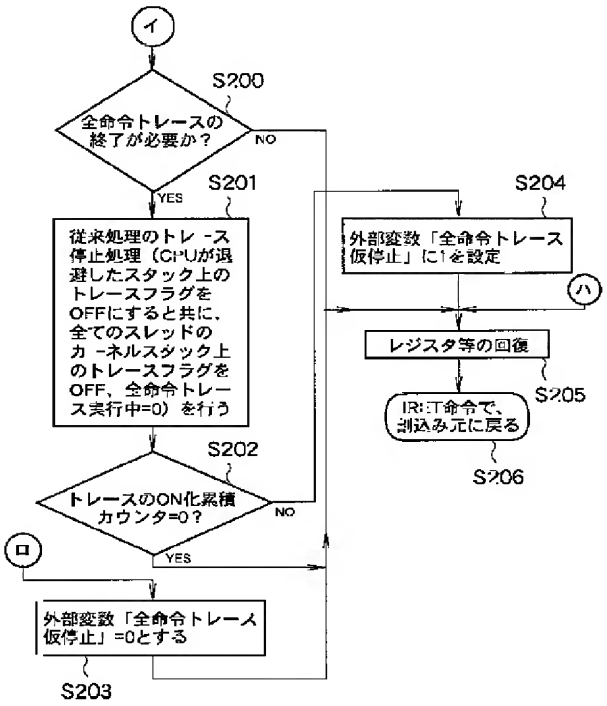
【図2】


50
スタック
51
旧スタックセグメント番号（4バイト：CPUが設定、割込みがユーザ空間で発生した場合のみ存在）
52
旧スタックポインタ（4バイト：CPUが設定、割込みがユーザ空間で発生した場合のみ存在）
53
旧状態フラグ（4バイト：CPUが設定）
54
旧コードセグメント番号（4バイト：CPUが設定）
55
旧命令アドレス（4バイト：CPUが設定）
56
レジスタ情報群（48バイト：ステップ100が設定）
60
ユーザの状態フラグ（4バイト）
70
スタックポインタ

【図4】


START
S100
レジスタ等の退避
S101
全命令トレース仮停止=0？
NO
YES
S102
全命令トレース実行中？
YES
NO
S103
オペレーティングシステムのデバッカへジャンプ
S104
全命令トレースデータを収集してバッファに格納し、適宜、2次記憶装置に出力
S105
シングルステップ割込みを起こした命令のアドレスはユーザー空間？
NO
YES
S106
シングルステップ割込みを起こした命令はPOPFD？
YES
NO
イ
S107
CPUによってスタックエリアに退避されたスタックポインタを得て、その旧スタックポインタがポイントするエリアの内容よりトレースフラグを得る
S108
上記トレースフラグはOFF？
NO
YES
S109
上記エリア内のトレースフラグをONに設定し、外部変数「トレースON化累積カウンタ」に1を加算する
S110
CPUが退避したスタック上のトレースフラグをOFFにし、外部変数「トレースON化累積カウンタ」より1を減算する
S111
結果は0？
NO
YES
ロ
ハ

【図5】


イ
S200
全命令トレースの終了が必要か？
NO
YES
S201
従来処理のトレース停止処理（CPUが退避したスタック上のトレースフラグをOFFにすると共に、全てのスレッドのカーネルスタック上のトレースフラグをOFF、全命令トレース実行中=0）を行う
S202
トレースのON化累積カウンタ=0？
NO
YES
ロ
外部変数「全命令トレース仮停止」=0とする
S203
S204
外部変数「全命令トレース仮停止」に1を設定
ハ
レジスタ等の回復
S205
IRET命令で、割込み元に戻る
S206